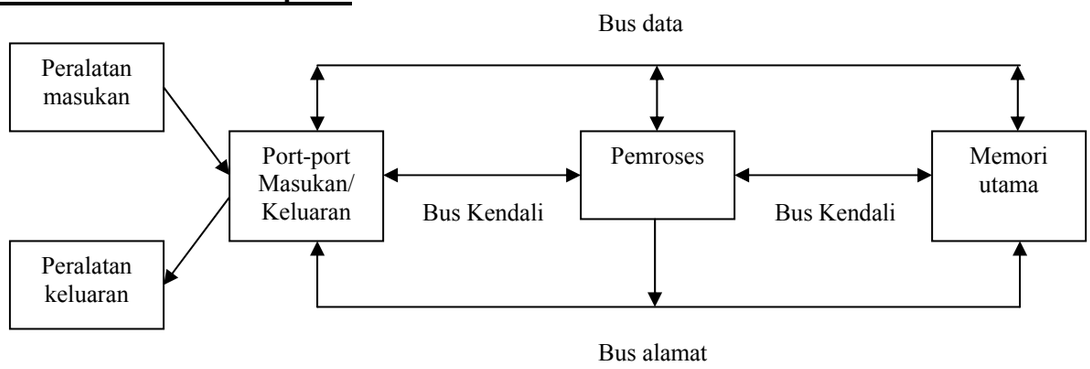
I. SKEMA DASAR SISTEM KOMPUTER

Pengampu : Idhawati Hestiningsih

Komponen sistem komputer

1. Pemroses (processor)
   Berfungsi mengendalikan operasi komputer & melakukan fungsi pemrosesan data.
2. Memori utama
   - Berfungsi menyimpan data & program
   - Biasanya volatile : tidak dapat mempertahankan data & program yang disimpan bila sumber daya energi (listrik) dihentikan.
3. Perangkat masukan dan keluaran
   Berfungsi memindahkan data antara komputer & lingkungan eksternal yaitu : perangkat penyimpan sekunder, perangkat komunikasi, terminal, dsb
4. Interkoneksi antarkomponen (bus)
   Adalah struktur & mekanisme untuk menghubungkan pemroses, memori utama, & perangkat masukan/keluaran.

Skema blok sistem komputer



PEMROSES

Berfungsi mengendalikan operasi komputer & melakukan fungsi pemrosesan data. Langkah-langkah yang dilakukan pemroses :

- mengambil instruksi yang dikodekan secara biner dari memori utama
- mendekode instruksi menjadi aksi-aksi sederhana
- melaksanakan aksi – aksi

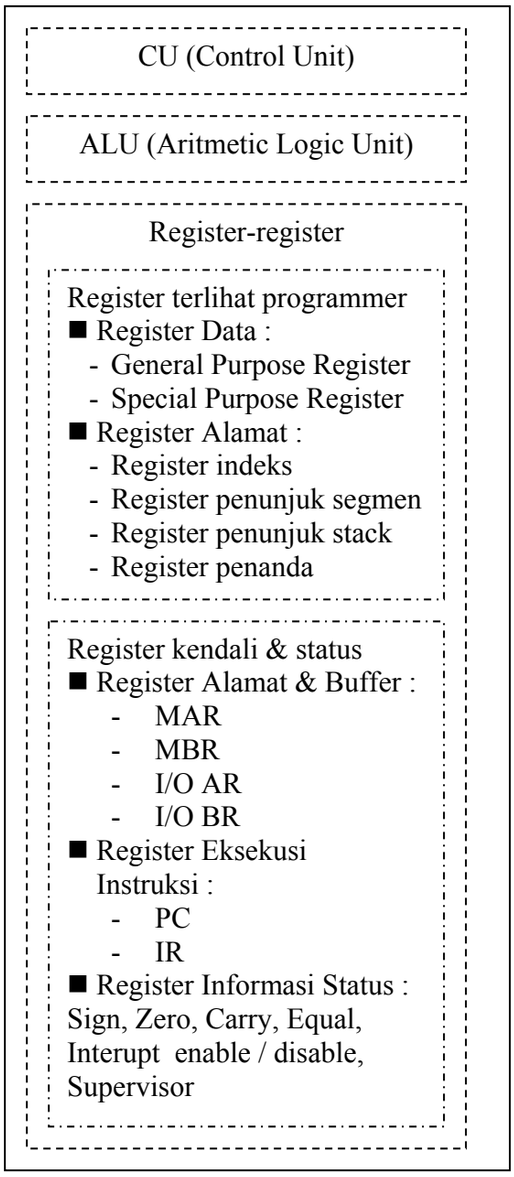
Operasi-operasi di komputer dapat dikategorikan menjadi 3 tipe, yaitu :

1. Operasi aritmatika : +,-,*, dsb
2. Operasi logika : OR, AND, XOR, inversi, dsb
3. Operasi pengendalian : percabangan, lompat, dsb

Pemroses terdiri dari :

- CU (Control Unit) : berfungsi mengendalikan operasi yang dilaksanakan sistem komputer.
- ALU (Aritmetic Logic Unit) : untuk komputasi yaitu melakukan operasi aritmatika & logika
- Register-register : berfungsi sebagai memori sangat cepat yang biasanya sebagai tempat operan-operan dari operasi yang akan dilakukan.

Skema blok Pemroses



Register dapat dikategorikan menjadi 2 :

1. **Register yg terlihat pemakai (pemrogram)**
   Pemrogram dapat memeriksa isi dari register-register tipe ini. Beberapa instruksi disediakan untuk mengisi (memodifikasi) register tipe ini. Terdiri dari 2 jenis :

1.1 Register Data : menyimpan suatu nilai untuk beragam keperluan

1.1.1 General purpose register
Digunakan untuk beraneka ragam keperluan pada suatu instruksi mesin yang melakukan suatu operasi terhadap data.

1.1.2 Special purpose register
Digunakan untuk menampung operasi floating point, menampung limpahan operasi penjumlahan atau perkalian.

1.2 Register Alamat : berisi alamat data di memori utama, alamat instruksi di memori utama,bagian alamat yang digunakan dalam penghitungan alamat lengkap

1.2.1 Register Indeks (index register)
Pengalamatan berindeks merupakan salah satu mode pengalamatan popular. Pengalamatan melibatkan penambahan indeks ke nilai dasar untuk memperoleh alamat efektif

1.2.2 Register penunjuk segmen (segment pointer register)
Pada pengalamatan bersegmen, memori dibagi menjadi segmen-segmen. Segmen berisi satu blok memori yang panjangnya dapat bervariasi.Untuk mengacu memori bersegmen digunakan pengacuan terhadap segmen dan offset di segmen itu. Register

penunjuk segmen mencatat alamat dasar (lokasi awal) dari segmen. Mode pengalamatan bersegmen sangat penting dalam manajemen memori.

1.2.3 Register penunjuk stack (stack pointer register)

Instruksi yang tak memerlukan alamat karena alamat operan ditunjuk register penunjuk stack. Operasi-operasi terhadap stack :

- instruksi **push** : menyimpan data pada stack, dengan meletakkan data di puncak stack
- instruksi **pop** : mengambil data dari puncak stack.

1.2.4 Register penanda (flag register)

Isi register merupakan hasil operasi dari pemroses. Register berisi kondisi-kondisi yang dihasilkan pemroses berkaitan dengan operasi yang baru saja dilaksanakan. Register ini terlihat oleh pemakai tapi hanya dapat diperbaharui oleh pemroses sebagai dampak (efek) operasi yang dijalankannya.

2. Register untuk kendali & status

Digunakan untuk mengendalikan operasi pemroses, kebanyakan tidak terlihat oleh pemakai. Sebagian dapat diakses dengan instruksi mesin yang dieksekusi dalam mode kontrol atau kernel sistem operasi.

2.1 Register untuk alamat dan buffer, terdiri dari :

2.1.1 MAR (Memory Address Register)

Untuk mencatat alamat memori yang akan diakses (baik yang akan ditulisi maupun dibaca)

2.1.2 MBR (Memory Buffer Register)

Untuk menampung data yang akan ditulis ke memori yang alamatnya ditunjuk MAR atau untuk menampung data dari memori (yang alamatnya ditunjuk oleh MAR) yang akan dibaca.

2.1.3 I/O AR (I/O Address Register)

Untuk mencatat alamat port I/O yang akan diakses(baik akan ditulisi / dibaca).

2.1.4 I/O BR (I/O Buffer Register)

Untuk menampung data yang akan dituliskan ke port yang alamatnya ditunjuk I/O AR atau untuk menampung data dari port (yang alamatnya ditunjuk oleh I/O AR) yang akan dibaca.

2.2 Register untuk eksekusi instruksi

2.2.1 PC (Program Counter) : mencatat alamat memori dimana instruksi di dalamnya akan dieksekusi

2.2.2 IR (Instruction Register) : menampung instruksi yang akan dilaksanakan

2.3 Register untuk informasi status

Register ini berupa satu register / kumpulan register. Kumpulan register ini disebut **PSW** (Program Status Word). PSW berisi kode-kode kondisi pemroses ditambah informasi-informasi status lain, yaitu :

- **Sign**
  Flag ini mencatat tanda yang dihasilkan operasi yang sebelumnya dijalankan
- **Zero**
  Flag ini mencatat apakah operasi sebelumnya menghasilkan nilai nol
- **Carry**
  Flag ini mencatat apakah dihasilkan carry (kondisi dimana operasi penjumlahan/ perkalian menghasilkan bawaan yang tidak dapat ditampung register akumulator)
- **Equal**
  Flag ini mencatat apakah operasi menghasilkan kondisi sama dengan
- **Interupt enable/disable**
  Flag ini mencatat apakah interrupt sedang dapat diaktifkan atau tidak
- **Supervisor**
  Flag ini mencatat mode eksekusi yang dilaksanakan, yaitu mode supervisor atau bukan. Pada mode supervisor maka seluruh instruksi dapat dilaksanakan sedang untuk mode bukan mode supervisor(mode user) maka beberapa instruksi kritis tidak dapat diaktifkan.

MEMORI

Memori berfungsi untuk menyimpan data dan program. Hirarki memori berdasarkan kecepatan akses :

Tercepat	Register	
	Chace memory	
	Main memory	
	Disk chace	
	Magnetik disc	
Terlambat	Magnetic tape	Optical disc

- Harga : semakin ke bawah, harga semakin murah, harga dihitung dari rasio rupiah per bit data disimpan
- Kapasitas : semakin ke bawah, kapasitas makin terbatas
- Kecepatan akses : semakin ke bawah, semakin lambat
- Frekuensi pengaksesan : semakin ke bawah, semakin rendah frekuensi pengaksesan

Setiap kali pemroses melakukan eksekusi, pemroses harus membaca instruksi dari memori utama. Agar eksekusi dilakukan secara cepat maka harus diusahakan instruksi tersedia di memori pada lapisan berkecepatan akses lebih tinggi. Kecepatan eksekusi ini akan meningkatkan kinerja sistem. Konsep ini diimplementasikan antara lain berupa :

- **Chace memory**
  Merupakan memori berkapasitas terbatas, berkecepatan tinggi yang lebih mahal dibanding memori utama. Chace memory adalah di antara memori utama dan register pemroses yang berfungsi agar pemroses tidak langsung mengacu memori utama tetapi di chace memory yang kecepatan akses lebih tinggi. Metode ini akan meningkatkan kinerja sistem.
- **Buffering**
  Bagian memori utama untuk menampung data yang akan ditransfer dari / ke perangkat masukan / keluaran dan penyimpan sekunder. Buffering dapat mengurangi frekuensi pengaksesan dari/ke perangkat masukan/keluaran dan penyimpan sekunder sehingga meningkatkan kinerja sistem.

PERANGKAT MASUKAN / KELUARAN

Perangkat masukan/keluaran terdiri dari 2 bagian :

1. Komponen mekanis : perangkat itu sendiri
2. Komponen elektronis : pengendali perangkat berupa chip controller

Perangkat adalah perangkat nyata yang dikendalikan chip controller di board system atau card. Controller dihubungkan dengan pemroses dan komponen-komponen lain lewat bus. Controller berbeda-beda, tapi biasanya mempunyai register-register untuk mengendalikannya.

INTERKONEKSI ANTAR KOMPONEN (BUS)

Bus terdiri dari 3 macam :

1. Bus alamat (address bus)
   CPU mengirim alamat lokasi memori atau port yang ingin ditulis atau dibaca di bus alamat.
2. Bus data (data bus)
   CPU dapat membaca & mengirim data dari/ke memori atau port. Banyak perangkat pada sistem yang dicantolkan ke bus data tapi hanya satu perangkat pada satu saat yang dapat memakainya.
3. Bus kendali (control bus)
   CPU mengirim sinyal-sinyal pada bus kendali untuk memerintahkan memori atau port.
   Sinyal bus kendali antara lain :
   - memory read : memerintahkan baca memori
   - memory write : memerintahkan penulisan memori
   - I/O read : memerintahkan baca port I/O
   - I/O write : memerintahkan melakukan penulisan memori

Contoh mekanisme pembacaan

Untuk membaca data suatu lokasi memori, CPU mengirim alamat memori yang dikehendaki melalui bus alamat kemudian mengirim sinyal memory read pada bus kendali.

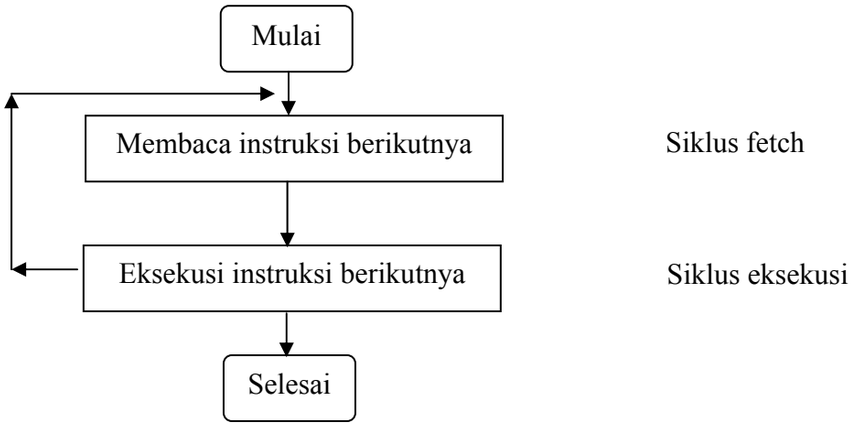
Sinyal memory read memerintahkan ke perangkat memori untuk mengeluarkan data pada lokasi tersebut ke bus data agar dibaca CPU.

EKSEKUSI INSTRUKSI

Tahap pemrosesan instruksi :

1. Pemroses membaca instruksi dari memori (fetch)
2. Pemroses mengeksekusi instruksi (execute)

Eksekusi program berisi pengulangan fetch dan execute. Pemrosesan 1 instruksi disebut satu siklus instruksi. Siklus eksekusi instruksi :



MODE EKSEKUSI INSTRUKSI

1. Mode pemakai (user mode)
   Mode dengan kewenangan rendah, program pemakai (aplikasi) biasa dieksekusi dalam mode ini.
2. Mode sistem (system mode)
   Mode dengan kewenangan tinggi. Biasanya rutin sistem atau kendali atau kernel dieksekusi dengan mode ini.

II. SISTEM OPERASI

Tujuan mempelajari sistem operasi :

1. Agar dapat merancang sendiri
2. Dapat memodifikasi sistem yang telah ada sehingga sesuai dengan kebutuhan
3. Dapat memilih di antara berbagai alternatif sistem operasi
4. Memaksimalkan penggunaan sistem operasi
5. Konsep dan teknik sistem operasi dapat diterapkan pada aplikasi-aplikasi lain.

Pengertian sistem operasi secara umum :

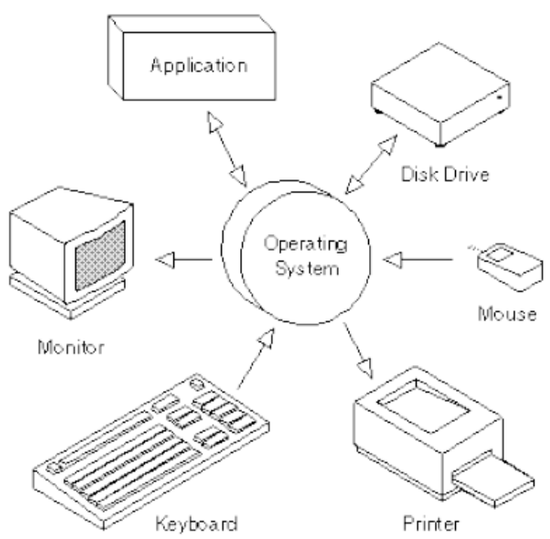
Sistem operasi sebagai *resource manager* yaitu pengelola seluruh sumber daya yang terdapat pada sistem komputer dan sebagai *extended machine* yaitu menyediakan sekumpulan layanan ke pemakai sehingga memudahkan dan menyamankan penggunaan serta pemanfaatan sumber daya sistem komputer.

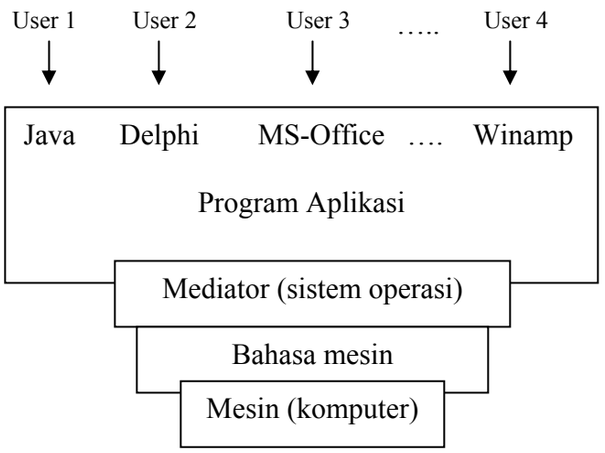
Prinsip dasar sistem operasi :

Sistem operasi merupakan program komputer yang berisi perintah-perintah (command) dan bertugas menjembatani pengertian manusia dengan komputer, sehingga komputer dapat bekerja sesuai keinginan.

Definisi sistem operasi :

1. Sistem operasi adalah software yang mengontrol hardware.
2. Program yang menjadikan hardware lebih mudah untuk digunakan
3. Kumpulan program yang mengatur kerja hardware sesuai keinginan user
4. Manager sumber daya atau pengalokasian sumber daya komputer, seperti mengatur memori, printer, dll
5. Sebagai program pengendali, yaitu program yang digunakan untuk mengontrol program yang lain
6. Sebagai kernel, yaitu program yang terus-menerus running selama komputer dihidupkan
7. Sebagai guardian yang menjaga komputer dari berbagai kejahatan komputer.





Kesimpulan :
Sistem operasi bertugas :
- sebagai perantara interaksi manusia & komputer
- mengatur hardware
- mengatur aplikasi

SEJARAH PERKEMBANGAN SISTEM OPERASI

1. Generasi pertama (tahun 1945 an – 1955 an)

Komputer elektronik digital pertama tidak memiliki sistem operasi. Sistem komputer diberi instruksi yang harus dikerjakan secara langsung sehingga semua operasi dilakukan secara manual dan hanya bisa digunakan untuk menghitung + - * /.

2. Generasi kedua (tahun 1955 – 1965)

Sistem komputer belum dilengkapi sistem operasi, tetapi beberapa fungsi dasar sistem operasi sudah ada, misal FMS (Fortran Monitoring System) dan IBSYS. Sistem komputer masa ini adalah *batch processing system*, yaitu pekerjaan (job) dikumpulkan dalam satu rangkaian kemudian dieksekusi secara berurutan. Tahun 1964, IBM mengeluarkan keluarga komputer System/360 yang dirancang agar kompatibel dengan banyak perangkat keras, menggunakan sistem operasi OS/360 dan berevolusi menjadi System 370.

3. Generasi ketiga (tahun1965 – 1980)

Sistem komputer dikembangkan untuk melayani banyak pemakai interaktif sekaligus dan online (secara langsung dihubungkan ke komputer). Sistem komputer menjadi multiuser dan multiprogramming.

Multiprogramming :

Komputer melayani banyak proses/job (program yang dijalankan) sekaligus pada satu waktu.

Cara yang dilakukan adalah dengan mempartisi memori menjadi beberapa bagian, dengan satu bagian memori untuk satu job berbeda. Saat satu job menunggu operasi masukan/keluaran selesai, job lain dapat menggunakan pemroses.

Karena komputer harus menanggapi permintaan-permintaan pemakai secara cepat dan bila tidak akan menyebabkan produktivitas pemakai menurun drastis, maka dikembangkan teknik *time sharing*, sehingga pemakai-pemakai merasa dilayani terus-menerus, padahal sebenarnya digilir per satuan waktu yang singkat.

Agar tidak terjadi bottleneck (kemacetan), dikembangkan teknik *spooling*, yaitu saat ada permintaan layanan peripheral, langsung diterima dan data disimpan lebih dulu di memori yang disediakan (berupa antrian), kemudian dijadwalkan agar secara nyata dilayani oleh peripheral.

Pada generasi ini muncul sistem operasi UNIX.

4. Generasi keempat (tahun 1980 – 199x)

Ditandai dengan meningkatnya kemampuan komputer dekstop (PC) dan teknologi jaringan TCP/IP. Pada generasi ini menuntut kenyamanan dalam mengoperasikan sistem komputer, yaitu dengan adanya GUI (graphical user interface = antarmuka komputer berbasiskan grafis yang nyaman). GUI dimulai dengan X Windows, kemudian Macintosh, Sun View dan MS Windows.

Pada 1990 dimulai era komputasi tersebar (distributed computing) dengan teknologi *distributed operating system* yaitu sistem operasi yang diperuntukkan jaringan komputer. Pemakai tak perlu menyadari keberadaan komputer-komputer yang terhubung, dimana pengalokasian kerja sudah secara otomatis dilaksanakan sistem operasi. Pemakai memandang jaringan komputer sebagai 1 uniprossesor besar, walau sebenarnya terdiri dari banyak prosessor (komputer) yang tersebar.

KOMPONEN DASAR SISTEM OPERASI

Kebanyakan sistem operasi memiliki komponen-komponen yang mendukung :

1. Manajemen proses
2. Manajemen memori utama
3. Manajemen berkas/file
4. Manajemen I/O
5. Manajemen penyimpanan sekunder
6. Jaringan
7. Sistem Proteksi
8. Command – Interpreter System

1. Manajemen Proses

- Proses adalah sebuah program yang sedang dijalankan (eksekusi).
  Suatu proses memerlukan sumberdaya pada saat eksekusi yaitu CPU time, memori, berkas dan peranti I/O.
- Sistem operasi bertanggung jawab terhadap aktifitas yang berhubungan dengan manajemen proses yaitu :
  - Pembuatan dan penghapusan proses
  - Penundaan dan pelanjutan proses
  - Penyedia mekanisme untuk :
    - Sinkronisasi antar proses
    - Komunikasi antar proses
    - Penanganan deadlock

2. Manajemen Memori Utama

- Memori sebagai tempat penyimpanan instruksi/data dari program.
- Penyimpanan yang cepat sehingga dapat mengimbangi kecepatan eksekusi instruksi CPU
- Terdiri dari“array words/bytes”yang besar
- Alamat digunakan untuk mengakses data (shared oleh CPU dan I/O devices)
- Umumnya main memory bersifat “volatile”–tidak permanen yaitu isinya akan hilang jika komputer di matikan.
- Sistem operasi bertanggung jawab untuk aktivitas berikutyangberhubungan dengan manajemen memori :
  - melacak pemakaian memori(siapa dan berapa besar?).
  - memilih program mana yang akan diload ke memori ketika bisa digunakan.
  - alokasi dan dealokasi memori sesuai yang dibutuhkan

3. Manajemen Berkas/File

- Berkas adalah kumpulan informasi yang berhubungan (sesuai dengan tujuan pembuat berkas tersebut).
- Biasanya berkas merepresentasikan program dan data.
- Sistem operasi bertanggungjawab :
  - pembuatan dan penghapusan berkas
  - pembuatan dan penghapusan direktori
  - mendukung manipulasi berkas dan direktori
  - memetakan berkas pada sistem sekunder
  - backup berkas pada media penyimpanan yang stabil (nonvolatile)

4. Manajemen I/O

Sistem I/O terdiri dari :

- Sistem buffer : menampung sementara data dari/ke peranti I/O
- Spooling : melakukan penjadwalan pemakaian I/O sistem supaya lebih efisien (antrian)
- Antarmuka devices-driver yang umum yaitu menyediakan device driver yang umum sehingga sistem operasi dapat seragam (buka, baca, tulis, tutup)
- Drivers untukspesifik perangkat keras spesifik
  Menyediakan driver untuk melakukan operasi rinci/detail untuk perangkat keras tertentu.

5. Manajemen Penyimpanan Sekunder

- Penyimpanan sekunder = penyimpanan permanen
- Karena memori utama bersifat sementara dan kapasitasnya terlalu kecil, maka untuk menyimpan semua data dan program secara permanen, sistem komputer harus menyediakan penyimpanan sekunder untuk dijadikan back-upmemori utama.
- Sistem operasi bertanggungjawab dalam aktivitas yang berhubungan dengan manajemen penyimpanan sekunder, yaitu :
    - manajemen ruang kosong
    - alokasi penyimpanan
    - penjadwalan disk

6. Jaringan (Sistem Terdistribusi)

- Sistem terdistribusi adalah kumpulan prosesor yang tidak berbagi memori atau clock. Setiap prosesor memiliki memori lokal masing-masing.
- Prosesor-prosesor dalam sistem terhubung dalam jaringan komunikasi.
- Sistem terdistribusi menyediakan akses pengguna ke bermacam-macam sumberdaya.
- Akses tersebut menyebabkan :
    - peningkatan kecepatan komputasi
    - peningkatan penyediaan data
    - peningkatan keandalan

7. Sistem Proteksi

- Proteksi berkenaan dengan mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, pengguna sistem maupun pengguna sumberdaya.
- Mekanisme proteksi harus :
    - membedakan antara penggunaan yang sah dan yang tidak sah.
    - spesifikasi kontrol untuk diterima
    - menyediakan alat untuk pemberlakuan sistem

8. Command – Interpreter System

- Sistem operasi menunggu instruksi dari pengguna (command driven).
- Program yang membaca instruksi dan mengartikan control statements (keinginan pengguna) umumnya disebut :
    - control-card interpreter
    - command-line interpreter
    - UNIX shell
- Command-interpreter system sangat bervariasi dari satu sistem operasi ke sistem operasi yang lain dan disesuaikan dengan tujuan dan teknologi peranti I/O yang ada.
  Contohnya: Windows, Pen-based (touch),dll.

STRUKTUR DASAR SISTEM OPERASI

Sistem operasi modern merupakan suatu sistem yang besar dan kompleks. Struktur sistem operasi merupakan komponen-komponen sistem operasi yang dihubungkan dan dibentuk di dalam kernel. Struktur sistem operasi yang pernah ada dan digunakan :

1. Sistem monolitik
2. Sistem berlapis
3. Sistem dengan mesin maya
4. Sistem dengan client – server
5. Sistem berorientasi objek

1. SISTEM MONOLITIK

Sistem operasi sebagai kumpulan prosedur dimana prosedur dapat saling dipanggil oleh prosedur lain di sistem bila diperlukan. Kernel berisi semua layanan yang disediakan sistem operasi untuk pemakai.

Kelemahan :

a. pengujian dan penghilangan kesalahan sulit karena tak dapat dipisahkan dan dilokalisasi
b. sulit dalam menyediakan fasilitas pengamanan
c. merupakan pemborosan bila setiap komputer harus menjalankan kernel monolitik sangat besar sementara sebenarnya tidak memerlukan seluruh layanan yang disediakan kernel. Tidak fleksibel
d. kesalahan pemrograman satu bagian dari kernel menyebabkan matinya seluruh sistem

Keunggulan :

Layanan dapat dilakukan sangat cepat karena terdapat di satu ruang alamat

UNIX sampai saat ini berstruktur monolitik. Meskipun monolitik, yaitu seluruh komponen/subsistem sistem operasi terdapat di satu ruang alamat tetapi secara rancangan adalah berlapis. Untuk mempermudah dalam pengembangan, pengujian & fleksibilitas UNIX menggunakan konsep kernel loadable modules, yaitu :

a. bagian-bagian kernel terpenting berada di memori utama secara tetap
b. bagian-bagian esensi lain berupa modul yang dapat ditambahkan ke kernel saat diperlukan dan dicabut begitu tidak digunakan lagi di waktu jalan (run-time).

2. SISTEM BERLAPIS

Sistem operasi dibentuk secara hirarki berdasar lapisan-lapisan, dimana lapisan-lapisan bawah memberi layanan lapisan lebih atas. Jadi lapisan n memberi layanan untuk lapisan n+1. Proses-proses di lapisan n dapat meminta layanan lapisan n – 1 untuk membangun layanan bagi lapisan n+1. Lapisan n dapat meminta layanan lapisan n – 1. Kebalikan tidak dapat, lapisan n tidak dapat meminta layanan n + 1. Masing-masing berjalan di ruang alamatnya sendiri.

Keunggulan :

Memiliki semua keunggulan rancangan modular, yaitu sistem dibagi menjadi beberapa modul dan tiap modul dirancang secara independen. Tiap lapisan dapat dirancang, dikode, dan diuji secara independen.

Kelemahan :

Fungsi – fungsi sistem operasi harus diberikan ke tiap lapisan secara hati-hati.

3. SISTEM dengan MESIN MAYA

Sistem operasi dapat menjalankan aplikasi-aplikasi untuk sistem operasi lain → konsep operating system emulator.

Contoh :

- MS Windows NT dapat menjalankan aplikasi untuk MS-DOS. Aplikasi tersebut dijalankan sebagai masukan bagi subsistem di MS-Windows NT yang mengemulsikan system calls yang dipanggil aplikasi dengan Win32API (system calls di MS Windows NT).
- Pengembang Linux membuat DOSEMU agar aplikasi untuk MS-DOS dapat dijalankan di Linux, WINE agar aplikasi untuk MS-Windows dapat dijalankan di Linux, iBCS agar aplikasi untuk Unix dapat dijalankan di Linux, dll

4. SISTEM dengan CLIENT-SERVER

Sistem operasi merupakan kumpulan proses dengan proses-proses dikategorikan sebagai server dan client.
Server : proses yang menyediakan layanan
Client : proses yang memerlukan/meminta layanan.

Proses client yang memerlukan layanan mengirim pesan ke server dan menanti pesan jawaban. Proses server setelah melakukan tugas yang diminta, mengirim pesan jawaban ke proses client. Server hanya menanggapi permintaan client dan tidak memulai percakapan dengan client.

Keunggulan :

- pengembangan dapat dilakukan secara modular
- kesalahan (bugs) di satu subsistem (diimplementasikan sebagai satu proses) tidak merusak subsistem-subsitem lain sehingga tidak mengakibatkan satu sistem mati secara keseluruhan.
- mudah diadaptasi untuk sistem tersebar

Kelemahan :

- layanan dilakukan lambat karena harus melalui pertukaran pesan
- pertukaran pesan dapat menjadi bottleneck

5. SISTEM BERORIENTASI OBJEK

Pada sistem berorientasi objek, layanan diimplementasikan sebagai kumpulan objek. Objek mengkapsulkan struktur data dan sekumpulan operasi pada struktur data itu. Tiap objek diberi tipe yang menandai properti objek seperti proses, direktori, berkas, dsb. Dengan memanggil operasi yang didefinisikan di objek, data yang dikapsulkan dapat diakses dan dimodifikasi. Model ini terstruktur dan memisahkan antara layanan yang disediakan dan implementasinya. Contoh sistem operasi berorientasi objek, antara lain : Eden, Choices, X-kernel, Medusa, Clouds, Amoeba, Muse, dsb.
Sistem operasi MS Windows NT telah mengadopsi beberapa teknologi berorientasi objek tapi belum keseluruhan.

III. PROSES

3.1 DESKRIPSI PROSES

- Proses adalah program yang sedang dieksekusi, termasuk didalamnya nilai-nilai dalam program counter, register, dan variabel-variabel yang ada.
- Perbedaan program dan proses :

	Program	Proses
Keberadaan	Ada secara nyata di ruang penyimpanan	Ada secara nyata dalam waktu yang terbatas
Sifat	Statis/pasif yang terletak di dalam suatu file	Dinamis/aktif karena dalam keadaan tereksekusi
Terdiri dari	Instruksi	Instruksi yang dieksekusi

- Sistem operasi mengontrol kejadian-kejadian yang ada dalam sistem komputer.
- Sistem operasi menjadwal dan mengirimkan proses untuk dieksekusi oleh proseseor, mengalokasikan sumber daya untuk proses dan menanggapi permintaan program pemakai untuk layanan-layanan dasar.

- Istilah-istilah yang berkaitan proses, antara lain :

Multiprogramming (multitasking)
Manajemen banyak proses pada satu pemroses (processor).
Komputer pribadi (workstation) adalah sistem pemroses tunggal yang menjalankan sistem operasi multiprogramming seperti MS Windows. Banyak proses dijalankan bersamaan, masing-masing proses mendapat bagian memori dan kendali tersendiri. Sistem operasi mengalih-alihkan pemroses di antara proses-proses tersebut. Program-program yang dijalankan sebenarnya bersifat :

- saling tak bergantung (independen)
- satu program pada satu saat (one program at any instant)

Multiprocessing
Manajemen banyak proses di komputer multiprocessor (banyak pemroses/processor di dalamnya). Dulunya sistem ini hanya terdapat di sistem besar, mainframe, dan minikomputer. Saat ini workstation telah dilengkapi multiprocessor.

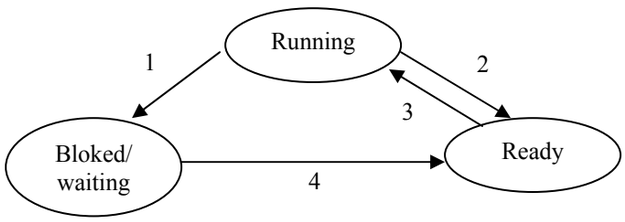
Distributed Processing
Manajemen banyak proses yang dieksekusi di banyak sistem komputer yang tersebar (terdistribusi).

3.2 DIAGRAM STATE PROSES

- Secara garis besar ada **3 state/status dasar/utama** pada proses :

State/Status	Deskripsi
Running	Pemroses sedang mengeksekusi instruksi proses tersebut
Ready	Proses siap (ready) dieksekusi, tapi pemroses belum tersedia untuk eksekusi proses ini = proses sedang menunggu jatah waktu dari pemroses untuk dieksekusi
Blocked(waiting)	Proses menunggu event/kejadian untuk melengkapi tugasnya. Contoh proses menunggu : • selesainya operasi perangkat masukan/keluaran • tersedianya memori • tibanya pesan jawaban

Hubungan ke-3 state dasar digambarkan dengan **diagram state proses** :



Keterangan perpindahan state :

1. Process blocks for input
   Terjadi pada saat sebuah proses menemukan bahwa dirinya tidak bisa dilanjutkan karena proses tidak mendapatkan apa yang diinginkan → input tidak tersedia
   Contoh : saat proses cetak (print), karena kertas/tinta habis maka proses cetak berhenti/menunggu

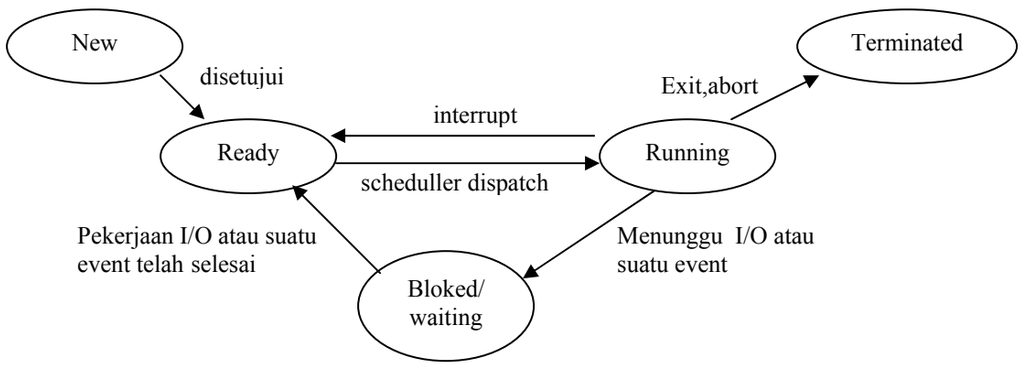
2. Scheduler picks another process
   Scheduler/penjadwal memutuskan bahwa sebuah proses sudah berjalan terlalu lama dan sudah waktunya untuk memanggil proses yang lain

3. Scheduler picks this process
   Proses-proses yang lain sudah mendapatkan jatahnya, dan tiba giliran proses yang tertunda untuk dijalankan.

4. Input becomes available
   Kejadian di luar yang sedang ditunggu sebuah proses sudah terlaksana yaitu misal sebuah proses yang menunggu input dari sebuah output proses yang lain. Bila tidak ada proses yang sedang berjalan, maka transisi 3 segera dilaksanakan dan proses segera berjalan/menunggu di state ready sampai CPU tersedia

   Misal : proses mencetak tadinya berhenti karena menunggu kertas dimasukkan, setelah kertas dimasukkan proses mencetak dapat dilanjutkan

- Diagram state dengan **5 status** : pengembangan dari 3 status



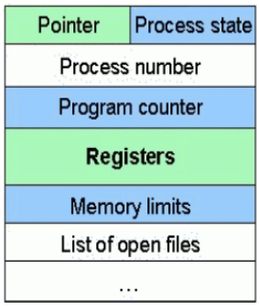
State/Status	Deskripsi
Running	Pemroses sedang mengeksekusi instruksi proses tersebut
Ready	Proses siap (ready) dieksekusi, tapi pemroses belum tersedia untuk eksekusi proses ini = proses sedang menunggu jatah waktu dari pemroses untuk dieksekusi
Blocked (waiting)	Proses menunggu event/kejadian untuk melengkapi tugasnya. Contoh proses menunggu : • selesainya operasi perangkat masukan/keluaran • tersedianya memori • tibanya pesan jawaban
New	Proses baru saja dibuat
Terminated	Proses telah selesai dieksekusi

- Hanya satu proses yang dapat berjalan pada prosesor mana pun pada satu waktu. Namun, banyak proses yang dapat berstatus *Ready* atau *Waiting*.

- Jadi ada tiga kemungkinan bila sebuah proses memiliki status *Running* :
  1. Jika program telah selesai dieksekusi maka status dari proses tersebut akan berubah menjadi *Terminated*.
  2. Jika waktu yang disediakan oleh OS untuk proses tersebut sudah habis maka akan terjadi *interrupt* dan proses tersebut kini berstatus *Ready*.
  3. Jika suatu event terjadi pada saat proses dieksekusi (seperti ada permintaan I/O) maka proses tersebut akan menunggu event tersebut selesai dan proses berstatus *Waiting*.

3.2.1 PCB (Process Control Block)

- Setiap proses digambarkan dalam sistem operasi oleh sebuah *process control block* (PCB) – juga disebut sebuah *control block*.



Gambar PCB

- PCB berisikan banyak bagian dari informasi yang berhubungan dengan sebuah proses yang spesifik,termasuk hal-hal di bawah ini:
  1. **Status Proses**
     Status *new*, *ready*, *running*, *waiting*, *terminated*, dan juga banyak lagi.
  2. ***Program Counter***
     Suatu stack yang berisi alamat berikutnya yang akan dieksekusi oleh proses tersebut
  3. **CPU register**
     Register tersebut termasuk *accumulator*, register indeks, *stack pointer*, *general-purposes register*, ditambah *code information* pada kondisi apa pun. Beserta dengan *program counter*, keadaan/status informasi harus disimpan ketika gangguan terjadi, untuk memungkinkan proses tersebut berjalan/bekerja dengan benar setelahnya
  4. **Informasi penjadwalan CPU**
     Informasi ini berisi prioritas dari suatu proses, pointer ke antrian penjadwalan, dan beberapa parameter penjadwalan yang lainnya.
  5. **Informasi manajemen memori**
     Informasi ini dapat termasuk suatu informasi sebagai nilai dari dasar dan batas register, tabel halaman, atau tabel segmen tergantung pada sistem memori yang digunakan oleh sistem operasi
  6. **Informasi pencatatan**
     Informasi ini termasuk jumlah dari CPU dan waktu riil yang digunakan, batas waktu, jumlah akun, jumlah *job* atau proses, dan banyak lagi.
  7. **Informasi status I/O**
     Informasi termasuk daftar dari perangkat I/O yang digunakan pada proses ini, daftar berkas-berkas yang sedang diakses dan banyak lagi.

- PCB hanya berfungsi sebagai tempat penyimpanan informasi yang dapat bervariasi dari proses yang satu dengan yang lain

- **Pembentukan Proses**
  Saat komputer berjalan, terdapat banyak proses yang berjalan secara bersamaan.
  Sebuah proses dibuat melalui *system call create-process* yang membentuk proses turunan (*child process*) yang dilakukan oleh proses induk (*parent process*).
  Proses turunan tersebut juga mampu membuat proses *Process Control Block* baru sehingga semua proses ini pada akhirnya membentuk pohon proses.
  Ketika sebuah proses dibuat maka proses tersebut dapat memperoleh sumber-daya seperti waktu CPU, memori, berkas, atau perangkat I/O. Sumber daya ini dapat diperoleh langsung dari sistem operasi, dari proses induk yang membagi-bagikan sumber daya kepada setiap

proses turunannya, atau proses turunan dan proses induk berbagi sumber-daya yang diberikan sistem operasi.

3.2.2 Operasi pada Proses

Sistem operasi dalam mengelola proses dapat melakukan operasi – operasi terhadap proses, diantaranya :

- penciptaan proses (create a process)
- penghancuran/terminasi proses (destroy a process)
- penundaan proses (suspend a process)
- pelanjutan kembali proses (resume a process)
- pengubahan prioritas proses
- memblock proses
- membangunkan proses
- menjadwalkan proses
- memungkinkan proses berkomunikasi dengan proses lain

a. Penciptaan dan penghancuran proses

Penciptaan proses melibatkan banyak aktivitas yaitu :

- menamai (memberi identitas proses)
- menyisipkan proses pada senarai proses atau tabel proses
- menentukan prioritas awal proses
- menciptakan PCB
- mengalokasikan sumber daya awal bagi proses

Kejadian yang dapat menyebabkan penciptaan proses :

- pada lingkungan batch, sebagai tanggapan atas pemberian satu kerja (job)
- pada lingkungan interaktif, ketika pemakai baru berusaha log on
- sebagai tanggapan suatu aplikasi, seperti permintaan pencetakan file, sistem operasi dapat menciptakan proses yang akan mengelola pencetakan itu.
- proses menciptakan proses lain (proses anak)

Penghancuran proses melibatkan pembebasan proses dari sistem, yaitu :

- sumber daya – sumber daya yang dipakai dikembalikan
- proses dihancurkan dari senarai atau tabel sistem
- PCB dihapus (ruang memori PCB dikembalikan ke pool memori bebas)

  Penghancuran lebih rumit bila proses telah menciptakan proses-proses lain. Terdapat dua pendekatan, yaitu :
  - pada beberapa sistem, proses-proses turunan dihancurkan saat proses induk dihancurkan secara otomatis
  - beberapa sistem lain menganggap proses anak independen terhadap proses induk sehingga proses anak tidak secara otomatis dihancurkan saat proses induk dihancurkan.

Alasan-alasan penghancuran proses (penyebab terminasi) :

1. Selesainya proses secara normal
   Proses mengeksekusi panggilan layanan sistem operasi untuk menandakan bahwa proses telah berjalan secara lengkap
2. Batas waktu terlewati
   Proses telah berjalan melebihi batas waktu total yang dispesifikasikan.
3. Memori tidak tersedia
   Proses memerlukan memori lebih banyak daripada yang dapat disediakan sistem.
4. Pelanggaran terhadap batas memori
   Proses mencoba mengakses lokasi memori yang tidak diijinkan diakses.
5. Terjadi kesalahan karena pelanggaran proteksi
   Proses berusaha menggunakan sumber daya atau file yang tidak diijinkan dipakainya, atau proses mencoba menggunakannya tidak untuk peruntukkannya, seperti menulis file read-only.
6. Terjadi kesalahan aritmatika
   Proses mencoba perhitungan terlarang, seperti pembagian dengan nol atau mencoba menyimpan angka yang lebih besar daripada yang dapat diakomodasi oleh perangkat keras.

7. Waktu telah kadaluwarsa
   Proses telah menunggu lebih lama daripada maksimum yang ditentukan untuk terjadinya suatu kejadian spesifik.
8. Terjadi kegagalan masukan/keluaran
   Kesalahan muncul pada masukan atau keluaran, seperti ketidakmampuan menemukan file, kegagalan membaca atau menulis setelah sejumlah maksimum percobaan yang ditentukan (misal : area rusak didapatkan pada disk, atau operasi tak valis seperti membaca dari line printer)
9. Instruksi yang tak benar
   Proses berusaha mengeksekusi instruksi yang tak ada.
10. Terjadi usaha memakai instruksi yang tak diijinkan
    Proses berusaha menggunakan instruksi yang disimpan untuk sistem operasi.
11. Kesalahan penggunaan data
    Tipe data yang digunakan salah atau tidak diinisialisasi.
12. Diintervensi oleh sistem operasi atau operator
    Untuk suatu alasan, operator atau sistem operasi mengakhiri proses (misal terdapat deadlock).
13. Berakhirnya proses induk
    Ketika parent berakhir, sistem operasi mungkin dirancang secara otomatis mengakhiri semua anak proses dari parent itu.
14. Atas permintaan dari proses induk
    Parent process biasanya mempunyai otoritas mengakhiri suatu anak proses.

b. Penundaan dan pelanjutan (pengaktifan) kembali proses

Penundaan (suspend) sering dilakukan sistem untuk memindahkan proses-proses tertentu guna mereduksi beban sistem selama beban puncak. Penundaan biasanya berlangsung singkat. Pengaktifan kembali (resuming) proses yaitu menjalankan proses dari titik (instruksi) dimana proses ditunda.

Operasi suspend dan resume penting karena :

1. Jika sistem berfungsi secara buruk dan mungkin gagal maka proses-proses dapat disuspend agar diresume setelah masalah diselesaikan.
   Contoh : saat proses pencetakan, bila tiba-tiba kertas habis maka proses disuspend. Setelah kertas dimasukkan kembali, proses pun dapat diresume.
2. Pemakai yang ragu mengenai hasil proses dapat mensuspend proses (bukan membuang (abort) proses). Saat pemakai yakin proses akan berfungsi secara benar maka dapat me-resume proses yang di-suspend.
3. Sebagai tanggapan terhadap fluktuasi jangka pendek beban sistem, beberapa proses dapat disuspend dan diresume saat beban kembali ke tingkat normal.

SYSTEM CALL

- **System call** menyediakan interface antara proses dengan sistem operasi. System call biasanya berbentuk instruksi bahasa assembly.
- **System call** = suatu kumpulan instruksi extended yang disediakan oleh sistem operasi yang berfungsi sebagai interface antara sistem operasi dengan program pemakai.
- Pogram pemakai berkomunikasi dengan sistem operasi dan meminta layanan dari sistem operasi dengan membuat system call.
- System call membuat, menghapus, dan menggunakan berbagai macam objek software yang dikelola oleh sistem operasi.

3.3 IMPLEMENTASI PROSES

3.3.1 Tabel Untuk Proses

Tiap proses mempunyai state yang perlu diperhatikan sistem operasi. Sistem operasi mencatat state proses dengan beragam tabel atau senarai yang saling berhubungan, yaitu :

- Tabel informasi manajemen memori
- Tabel informasi manajemen masukan/keluaran
- Tabel informasi sistem file
- Tabel proses

Tabel informasi manajemen memori

Merupakan tabel untuk menjaga keutuhan memori utama dan memori sekunder, memuat informasi :

- alokasi memori utama yang dipakai proses
- alokasi memori sekunder yang dipakai proses (bila menggunakan manajemen memori dengan swapping)
- atribut segmen memori utama dan sekunder
- informasi lain yang digunakan untuk pengelolaan memori

Tabel informasi manajemen masukan/keluaran

Untuk mengelola perangkat masukan/keluaran. Saat perangkat masukan/keluaran digunakan proses tertentu, perlu dijaga agar proses lain tidak memakainya. Sistem operasi perlu mengetahui status operasi masukan/keluaran dan lokasi memori utama yang digunakan untuk transfer data.

Tabel informasi sistem file

Berisi informasi mengenai ekstensi file, lokasi pada memori sekunder, status saat itu dan menyimpan atribut-atribut file lain.

Tabel proses

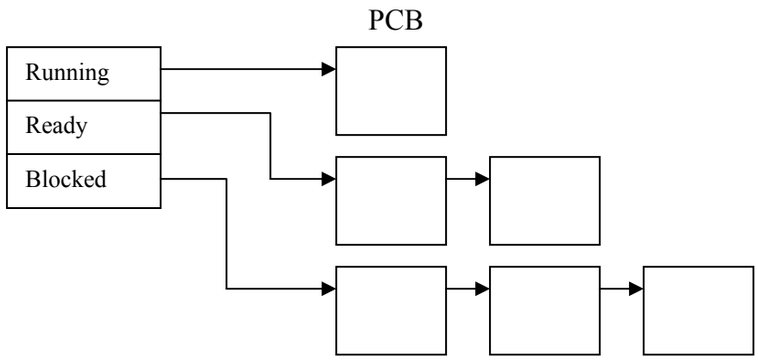
Mengelola informasi proses di sistem operasi, lokasinya di memori. Tabel juga berisi status dan atribut proses yang lain

Proses ditempatkan di memori utama di lokasi tertentu, proses mempunyai satu ruang alamat tersendiri. Ruang alamat yang digunakan proses disebut citra proses (process image). Tabel berikut menunjukkan isi dari citra proses.

Elemen citra proses	Deskripsi
Data pemakai	Bagian yang dapat dimodifikasi berupa data program, daerah stack pemakai
Program pemakai	Program biner yang (akan) dieksekusi
Stack sistem	Digunakan untuk menyimpan parameter dan alamat pemanggilan untuk prosedur dan system calls
PCB (Program Control Block)	Berisi informasi yang diperlukan oleh sistem operasi dalam mengendalikan proses.

3.3.2 PCB dan Senarai Proses

Tiap PCB berisi informasi mengenai proses yang diperlukan sistem operasi. PCB dibaca dan atau dimodifikasi rutin sistem operasi seperti penjadwalan, alokasi sumber daya, pemrosesan interupsi, monitoring, analisis kinerja. Kumpulan PCB mendefinisikan state sistem operasi. Untuk menyatakan senarai proses di sistem operasi dibuat senarai PCB.



Gambar diatas memperlihatkan hanya 1 PCB berada di senarai running. PCB ini menyatakan proses yang saat itu sedang dieksekusi pemroses sehingga hanya satu proses yang running.
Proses ready digambarkan dengan PCB proses-proses di senarai ready. Proses-proses menunggu dijadwalkan untuk dieksekusi pemroses. Proses yang dijadwalkan dieksekusi (yaitu mengalami transisi dari state ready menjadi running) maka PCB-nya dipindah dari senarai ready ke senarai running.
Proses running (PCB-nya berada di senarai running) dipindah sesuai state yang dialami proses proses itu :

- Bila proses berakhir (selesai) maka dijalankan operasi terminasi sehingga PCB-nya tak ada lagi.
- Bila proses di-blocked karena menunggu alokasi sumber daya maka PCB-nya dipindah ke senarai blocked
- Bila proses dijadwalkan habis jatah waktu eksekusinya maka PCB-nya dipindahkan ke senarai ready.

Proses yang sedang blocked berpindah menjadi ready bila sumber daya yang ditunggu telah teralokasi untuknya. Untuk itu PCB-nya dipindahkan ke senarai ready.

3.3.3 Pengaksesan Informasi di PCB

Rutin-rutin sistem operasi perlu mengakses informasi di PCB. Tiap proses dilengkapi ID unik yang digunakan sebagai indeks (penunjuk) ke tabel untuk mengambil PCB.
Untuk menghindari bug (kesalahan pemrograman) pada rutin tunggal yang bisa mengakibatkan rusaknya PCB dan menghindari perubahan rancangan struktur dan semantiks PCB, maka semua rutin sistem operasi melewati satu rutin khusus yaitu rutin penanganan PCB dalam mengakses PCB. Tugas rutin adalah memproteksi PCB dan menjadi perantara pembacaan dan penulisan PCB.
Kelemahannya : adanya overhead kinerja karena harus memanggil rutin penanganan PCB. Pengaksesan langsung terhadap PCB tentu lebih cepat daripada harus memanggil rutin penanganan PCB.

3.4 TAHAP-TAHAP PENCIPTAAN PROSES

Tahap penciptaan proses :

1. Beri satu identifier unik ke proses baru. Isian baru ditambahkan ke tabel proses utama yang berisi satu isian per proses.
2. Alokasikan ruang untuk proses
3. PCB harus diinisialisasi
4. Kaitan-kaitan antar tabel dan senarai yang cocok dibuat
5. Bila diperlukan struktur data lain maka segera buat struktur data itu.

IV. PENJADWALAN PROSES

4.1 DEFINISI

Penjadwalan merupakan kumpulan kebijaksanaan dan mekanisme di sistem operasi yang berkaitan dengan urutan kerja yang dilakukan sistem komputer. Penjadwalan bertugas memutuskan proses yang harus berjalan dan kapan atau berapa lama proses itu berjalan.
Sasaran utama penjadwalan proses adalah optimasi kinerja menurut kriteria tertentu, yaitu :

- adil
- efisiensi
- waktu tanggap (response time)
- turn arround time
- throughput

Adil
Proses-proses diperlakukan sama yaitu mendapat jatah waktu pemroses yang sama dan tak ada proses yang tak kebagian layanan pemroses sehingga mengalami *starvation.*

Efisiensi
Efisiensi atau utilisasi pemroses dihitung dengan perbandingan (rasio) waktu sibuk pemroses. Sasaran penjadwalan adalah menjaga agar pemroses tetap dalam keadaan sibuk sehingga efisiensi mencapai maksimum. Sibuk adalah pemroses tidak menganggur, termasuk waktu yang dihabiskan untuk mengeksekusi program pemakai dan sistem operasi.

Waktu tanggap (response time)
Waktu tanggap pada sistem interaktif
Adalah waktu yang dihabiskan dari saat karakter terakhir dari perintah dimasukkan atau transaksi sampai hasil pertama muncul di layar (terminal) → disebut terminal response time

Waktu tanggap pada sistem waktu nyata (real-time)
Adalah waktu dari saat kejadian (internal atau eksternal) sampai instruksi pertama rutin layanan yang dimaksud dieksekusi → disebut event response time

Turn arround time
Adalah waktu yang dihabiskan dari saat program atau job mulai masuk ke sistem sampai proses diselesaikan sistem. Waktu yang dimaksud adalah waktu yang dihabiskan di dalam sistem.

turn arround time = waktu eksekusi + waktu menunggu

Sasaran penjadwalan adalah meminimalkan turn arround time.

Throughput
Adalah jumlah kerja atau jumlah job yang dapat diselesaikan dalam satu unit waktu. Sasaran penjadwalan adalah memaksimalkan jumlah job yang diproses per satu interval waktu. Lebih tinggi angka throughput, lebih banyak kerja yang dilakukan sistem.

4.2 TIPE-TIPE PENJADWALAN

Tiga tipe penjadwal dapat berada secara bersama-sama pada sistem operasi yang kompleks, yaitu :

1. Penjadwal jangka pendek (short – term scheduller)
2. Penjadwal jangka menengah (medium – term scheduller)
3. Penjadwal jangka panjang (long– term scheduller)

Penjadwal jangka pendek
Tugas menjadwalkan alokasi pemroses diantara proses-proses ready di memori utama.

Penjadwal jangka menengah
Setelah eksekusi selama satu waktu, proses mungkin ditunda karena membuat permintaan layanan masukan/keluaran atau memanggil suatu system call. Proses-proses tertunda tidak dapat membuat suatu kemajuan menuju selesai sampai kondisi-kondisi yang menyebabkan tertunda dihilangkan. Agar ruang memori dapat bermanfaat, maka proses dipindah dari memori utama ke memori sekunder agar tersedia ruang untuk proses-proses lain. Kapasitas memori utama terbatas untuk

sejumlah proses aktif. Aktifitas pemindahan proses yang tertunda dari memori utama ke memori sekunder disebut *swapping*.

Penjadwal jangka panjang
Penjadwal jangka panjang bekerja terhadap antrian batch dan memilih batch berikutnya yang harus dieksekusi. Batch biasanya adalah proses-proses dengan penggunaan sumber daya yang intensif (yaitu waktu pemroses, memori, perangkat masukan/keluaran), program-program ini berprioritas rendah, digunakan sebagai pengisi (agar pemroses sibuk) selama periode aktivitas job-job interaktif rendah.Sasaran utama penjadwal jangka panjang adalah memberi keseimbangan job-job campuran.

4.3 STRATEGI PENJADWALAN

Ada 2 strategi penjadwalan :

- Penjadwalan nonpreemptive
- Penjadwalan preemptive

Penjadwalan nonpreemptive
Proses yang sedang berjalan tidak dapat disela. Sekali proses berada di status running (sedang berjalan), maka proses tersebut akan dieksekusi terus sampai proses berhenti karena selesai atau diblok untuk menunggu I/O atau untuk meminta beberapa layanan dari sistem operasi; dan CPU tidak dapat diambil alih oleh proses yang lain.

Penjadwalan preemptive
Proses yang sedang berjalan dapat diinterupsi dan dipindah ke status ready oleh sistem operasi sehingga CPU dapat diambil alih proses yang lain.

4.4 ALGORITMA PENJADWALAN

Terdapat banyak algoritma, diantaranya :

a. Algoritma menggunakan strategi nonpreemptive
   - FIFO (First-in, First-out) atau FCFS (First-come, First-serve)
   - SJF (Shortest Job First)
   - HRN (Highest – Ratio Next)
b. Algoritma menggunakan strategi preemptive
   - MFQ (Multiple Feedback Queues)
   - RR (Round Robin)
   - SRF (Shortest Remaining First)
   - PS (Priority Schedulling)
   - GS (Guaranteed Schedulling)

Klasifikasi lain berdasarkan adanya prioritas di proses-proses tersebut, yaitu :

♦ Algoritma penjadwalan tanpa prioritas
♦ Algoritma penjadwalan berprioritas :
   - algoritma penjadwalan berprioritas statik
   - algoritma penjadwalan berprioritas dinamis

4.4.1 Penjadwalan FIFO (First in, First Out)

Merupakan penjadwalan nonpreemptive dan penjadwalan tidak berprioritas.
Penjadwalan FIFO adalah penjadwalan paling sederhana, yaitu :

- Proses-proses diberi jatah waktu pemroses berdasarkan waktu kedatangan
- Saat proses mendapat jatah waktu pemroses, proses dijalankan sampai selesai

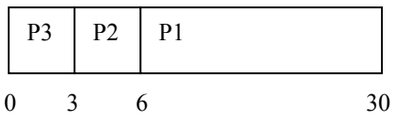
Penjadwalan ini adil yaitu proses yang datang duluan, dilayani duluan juga. Dikatakan tidak adil karena job-job yang perlu waktu lama membuat job-job pendek menunggu. Job-job tak penting dapat membuat job-job penting menunggu.
Contoh :
Misal ada tiga proses P1, P2, P3 yang datang dengan lama waktu kerja CPU (CPU Burst-time) masing-masing sbb :

Proses	Burst-time
P1	24
P2	3
P3	3

Jika proses datang dengan urutan P1, P2, P3 dan dilayani dengan algoritma FIFO maka dapat digambarkan Gantt Chart-nya :

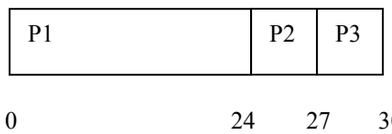
Dari Gantt Chart dapat diambil kesimpulan waktu tunggu untuk P1 adalah 0 milidetik, waktu tunggu untuk P2 adalah 24 milidetik, waktu tunggu P3 adalah 27 milidetik. Jadi rata-rata waktu tunggu (Average Waiting Time / AWT) adalah (0+24+27)/3 = 17 milidetik.

Kemudian jika waktu kedatangan proses adalah P3, P2, P1 maka Gantt Chartnya adalah



Dengan urutan kedatangan seperti diatas AWT = (0+3+6)/3 = 3 milidetik

Menentukan Turn Around Time dengan FIFO berdasarkan contoh diatas :



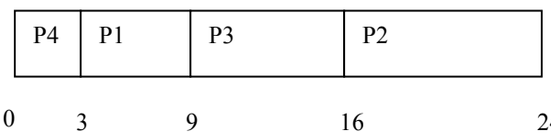
Turn around time (waktu penyelesaian) P1 adalah 24, P2 = 27, P3 = 30, maka rata-rata turn around time = (24+27+30)/3 = 27 milidetik

4.4.2 Penjadwalan SJF (Shortest Job First) → jarang digunakan

Merupakan penjadwalan nonpreemptive dan penjadwalan tidak berprioritas. Penjadwalan ini mengasumsikan waktu jalan proses (sampai selesai) diketahui sebelumnya. Mekanisme penjadwalan adalah menjadwalkan proses dengan waktu jalan terpendek lebih dulu sampai selesai.
Contoh :
Misal terdapat kumpulan proses dengan burst-time sebagai berikut :

Proses	Burst-time
P1	6
P2	8
P3	7
P4	3

Gantt Chart algoritma SJF :



Dari Gantt Chart dapat diambil nilai waktu tunggu untuk :
P1 = 3 milidetik
P2 = 16 milidetik
P3 = 9 milidetik
P4 = 0 milidetik

Jadi AWT yang dihasilkan adalah (3+16+9+0) / 4 = 7 milidetik.

Contoh :
Menentukan turn around time

Proses	Saat Tiba	Lama Proses
A	0	11
B	0	8
C	0	10
D	0	3
E	0	5

Proses	Saat Tiba	Lama Proses	Saat Mulai	Saat Selesai	Turn Around Time
D	0	3	0	3	3
E	0	5	3	8	8
B	0	8	8	16	16
C	0	10	16	26	26
A	0	11	26	37	37
				Jumlah Rata-rata	90 18

4.4.3 Penjadwalan HRN (Highest – Ratio Next)

Penjadwalan ini merupakan penjadwalan non-preemptive dan penjadwalan berprioritas dinamis, yang sebenarnya merupakan perbaikan dari algoritma SJF, karena pada SJF ada kemungkinan proses dengan waktu layanan lama akan menunggu lama untuk dieksekusi. Hal ini disebabkan adanya proses-proses lain yang waktu layanannya selalu lebih pendek dari proses tersebut.
Prioritas dinamis HRN dihitung berdasarkan rumus :

Prioritas = (waktu tunggu + waktu layanan) / waktu layanan

Proses dengan prioritas yang tertinggi akan dipilih untuk dieksekusi selanjutnya.

4.4.4 Penjadwalan MFQ (Multiple Feedback Queues)

Merupakan penjadwalan dengan banyak antrian, merupakan penjadwalan preemptive (by-time), penjadwalan berprioritas dinamis.
Penjadwalan ini untuk mencegah banyaknya swapping dengan proses-proses yang sangat banyak menggunakan pemroses (karena menyelesaikan tugasnya memakan waktu lama) diberi jatah waktu (jumlah kwanta/jumlah quantum) lebih banyak dalam satu waktu.
Penjadwalan ini menghendaki kelas-kelas prioritas bagi proses-proses yang ada. Kelas tertinggi berjalan selama satu kwanta, kelas berikutnya berjalan selama 2 kwanta,kelas berikutnya berjalan 4 kwanta, dst. Ketentuan yang berlaku :

1. Jalankan proses pada kelas tertinggi
2. Jika proses menggunakan seluruh kwanta yang dialokasikan maka diturunkan kelas prioritasnya
3. Proses yang masuk untuk pertama kali ke sistem langsung diberi kelas tertinggi.

4.4.5 Penjadwalan RR (Round Robin)

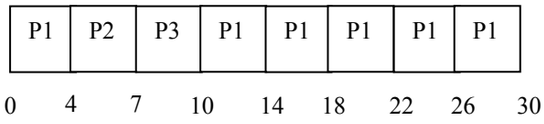
Merupakan penjadwalan preemptive, penjadwalan tanpa prioritas. Semua proses dianggap penting dan diberi sejumlah waktu pemroses yang disebut kwanta (quantum) atau time-slice dimana proses itu berjalan.
Ketentuan algoritma RR :

1. Jika quantum habis dan proses belum selesai maka proses menjadi runnable dan pemroses dialihkan ke proses lain
2. jika quantum belum habis dan proses menunggu suatu kejadian (selesainya I/O), maka proses menjadi blocked dan pemroses dialihkan ke proses lain.
3. jika quantum belum habis tapi proses telah selesai maka proses diakhiri dan pemroses dialihkan ke proses lain.

Contoh :
Misal kumpulan proses datang pada waktu 0 dengan spesifikasi :

Proses	**Burst-time**
P1	24
P2	3
P3	3

Jika digunakan quantum time 4 milidetik, maka proses P1 mendapat 4 milidetik yang pertama, 20 milidetik berikutnya akan disela oleh proses P2 dan P3 secara bergantian, sehingga Gantt Chart-nya dapat digambarkan sbb:



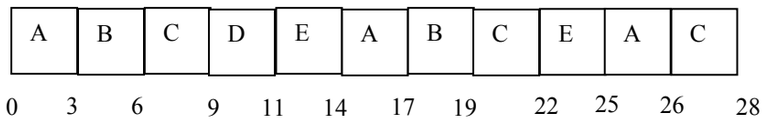
Waktu tunggu untuk tiap-tiap proses :

Proses	**Waiting Time**
P1	0+(10-4)=6
P2	4
P3	7

AWT yang terjadi adalah (6+4+7)/3 = 5,66 milidetik

Contoh :
Menentukan Turn Around Time untuk quantum waktu (q) = 3

Proses	Saat Tiba	Lama Proses
A	0	7
B	0	5
C	0	8
D	0	2
E	0	6



Proses	Saat Tiba	Lama Proses	Saat Mulai	Saat Selesai	Turn Around Time
A	0	7	0	26	26
B	0	5	3	19	19
C	0	8	6	28	28
D	0	2	9	11	11
E	0	6	11	25	25
				Jumlah Rata-rata	109 21,8

4.4.6. Penjadwalan SRF (Shortest Remaining First) → Penjadwalan sisa waktu terpendek, duluan

Merupakan penjadwalan preemptive, berprioritas dinamis. Pada SRF, proses dengan sisa waktu jalan diestimasi terendah dijalankan, termasuk proses-proses yang baru tiba.

1. Pada SJF begitu proses dieksekusi, proses dijalankan sampai selesai
2. Pada SRF proses yang sedang berjalan (running) dapat diambil alih proses baru dengan sisa jalan yang diestimasi lebih rendah.

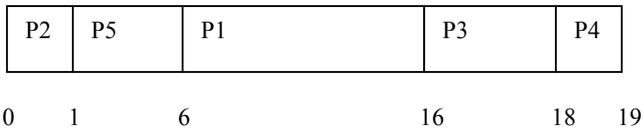
4.4.7 Penjadwalan PS (Priority Schedulling)

Tiap proses dilengkapi dengan prioritas. CPU dialokasikan untuk proses yang memiliki prioritas paling tinggi. Jika beberapa proses memiliki prioritas yang sama, maka akan digunakan algoritma FIFO. Contoh :

Jika ada 5 proses P1, P2, P3, P4 dan P5 dengan CPU burst :

Proses	Burst Time	Prioritas
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Gantt Chart :



Waktu tunggu untuk tiap-tiap proses :

Proses	Waiting Time
P1	6
P2	0
P3	16
P4	18
P5	1

Sehingga AWT = (6+0+16+18+1) = 8,2 ms.

Prioritas biasanya menyangkut masalah : waktu, memori yang dibutuhkan, banyaknya file yang boleh dibuka, dan perbandingan antara rata-rata I/O burst dengan rata-rata CPU burst.

Priority schedulling bersifat preemptive atau nonpreemptive. Jika ada proses P1 yang datang pada saat P0 sedang berjalan, maka akan dilihat prioritas P1. Seandainya prioritas P1 lebih besar dibanding dengan prioritas P0 maka pada non preemptive, algoritma tetap akan menyelesaikan P0 sampai habis CPU burst-nya, dan meletakkan P1 pada posisi head queue. Sedangkan pada preemptive, P0 akan dihentikan dulu, dan CPU ganti dialokasikan untuk P1.

4.4.8 Penjadwalan GS (Guaranteed Schedulling)

Penjadwalan ini harus menjamin bahwa algoritma tersebut mempunyai kinerja yang cukup bagus dan menjanjikan kelangsungan hidup yang baik. Contoh : misal ada n user yang sedang login, maka tiap-tiap user dijanjikan akan menerima 1/n dari kemampuan CPU.

Untuk meyakinkan bahwa setiap user mendapatkan jatah waktu menggunakan CPU sesuai dengan haknya maka sistem harus tahu berapa CPU time yang diperlukan oleh setiap proses dalam 1 user. Dan juga CPU time yang diperlukan oleh tiap-tiap user. Misal ada 5 user seperti tabel berikut :

User	CPU Time
A	5
B	4
C	8
D	1
E	2

Total waktu yang dibutuhkan untuk mengakses kelima user tersebut adalah 20 ms, sehingga diharapkan tiap user mendapatkan 20/5 = 4 ms. Pada kenyataannya, mulai dari login hingga saat ini tiap-tiap user telah mendapatkan CPU seperti terlihat pada tabel berikut. Dan rasio antara CPU yang diperoleh sampai saat ini (aktual) dengan CPU yang seharusnya diperoleh (4ms) dapat dicari :

User	CPU aktual	Rasio
A	3	¾ = 0.75
B	6	6/4=1.5
C	2	2/4=0.5
D	1	¼=0.25
E	1	¼=0.25

Dapat dilihat bahwa user A memiliki rasio 0.75, artinya A baru mendapatkan ¾ dari jatah waktu yang seharusnya diterima. User B memiliki rasio 1.5, artinya B telah mendapatkan 1.5 waktu dari yang seharusnya ia dapatkan. Algoritma ini kemudian akan menjalankan proses dengan rasio yang paling rendah dahulu hingga proses tersebut mendapatkan rasio melebihi rasio proses yang sebelumnya punya rasio satu tingkat lebih tinggi darinya.

V. KONGKURENSI (KEBERSAMAAN)

PERSAINGAN DAN KERJASAMA ANTAR PROSES

Persaingan antar proses terjadi ketika beberapa proses akan menggunakan sumber daya yang sama. Tidak ada pertukaran informasi diantara proses-proses yang sedang bersaing, tetapi eksekusi dari satu proses dapat menimbulkan akibat bagi proses-proses lain yang sedang bersaing.
Jika ada 2 proses yang akan mengakses ke suatu sumber daya tunggal, kemudian satu proses dialokasikan ke sumber daya tersebut oleh sistem operasi, maka proses yang lainnya akan menunggu. Pada kasus yang ekstrim, proses yang menunggu tersebut ada kemungkinan tidak akan pernah mendapatkan akses ke sumber daya sehingga tidak akan pernah selesai dengan sempurna. Hal ini juga terjadi akibat antar proses yang saling tidak peduli.

Kongkurensi merupakan landasan umum perancangan sistem operasi. Proses-proses disebut kongkuren jika proses-proses(lebih dari 1 proses) berada pada saat yang sama. Proses-proses kongkuren dapat sepenuhnya tak bergantung dengan lainnya tapi dapat juga saling berinteraksi/kerjasama. Proses-proses yang berinteraksi memerlukan sinkronisasi/koordinasi agar terkendali dengan baik.

PRINSIP-PRINSIP KONGKURENSI

Kongkurensi meliputi hal-hal berikut :

- alokasi waktu pemroses untuk proses-proses
- pemakaian bersama & persaingan untuk mendapatkan sumber daya
- komunikasi antar proses
- sinkronisasi aktivitas banyak proses

Kongkurensi dapat muncul pada 3 konteks yang berbeda, yaitu :

1. Banyak aplikasi (multiple application)
   Multiprogramming memungkinkan banyak proses sekaligus sehingga terdapat banyak aplikasi yang dijalankan pada sistem komputer. Banyak proses ini juga sangat berguna untuk sistem komputer tunggal (single user) karena sambil menunggu proses selesainya layanan (misal transfer berkas oleh modem atau pencetakan oleh printer) pemakai dapat berinteraksi dengan aplikasi lain seperti aplikasi game atau mengetik pada text editor.
2. Aplikasi terstruktur
   Perluasan prinsip-prinsip perancangan modular dan pemrograman terstruktur adalah suatu aplikasi dapat secara efektif diimplementasikan sebagai sekumpulan proses aplikasi. Dengan sekumpulan proses, maka tiap proses mempunyai satu layanan spesifik yang ditentukan.
3. Strukturisasi sistem operasi
   Keunggulan-keunggulan strukturisasi dapat juga diterapkan ke pemrograman sistem. Beberapa sistem operasi aktual yang dipasarkan dan dalam riset telah diimplementasikan sebagai sekumpulan proses.

Masalah yang dihadapi dengan proses kongkuren pada multiprogramming dan multiprocessing adalah serupa, yaitu bahwa kecepatan eksekusi proses-proses yang terdapat pada sistem tidak dapat diprediksi. Dengan demikian beragam kemungkinan yang dapat terjadi tidak dapat diprediksi.
Beberapa kesulitan yang dapat muncul diantaranya :

1. Pemakaian bersama sumberdaya global
2. Pengelolaan alokasi sumberdaya agar optimal
3. Pencarian kesalahan pemrograman

Hal-hal yang harus ditangani sistem operasi dengan adanya proses kongkuren :

1. Sistem operasi harus mengetahui proses-proses yang aktif
2. Sistem operasi harus alokasi dan dealokasi beragam sumberdaya untuk tiap proses aktif
3. Sistem operasi harus memproteksi data dan sumberdaya fisik masing-masing proses dari gangguan proses-proses lain
4. Hasil-hasil proses harus independen terhadap kecepatan relatif proses-proses lain dimana eksekusi dilakukan.

MUTUAL EXCLUSION

Mutual exclusion adalah jaminan hanya satu proses yang mengakses sumber daya pada suatu interval waktu tertentu, sedangkan proses lain dilarang mengerjakan hal yang sama → contoh : sumberdaya printer hanya bisa diakses 1 proses, tidak bisa bersamaan → sumber daya ini disebut sumber daya kritis

Bagian program yang sedang mengakses memori atau sumberdaya yang dipakai bersama disebut **critical section / region**. Kesuksesan proses-proses kongkuren memerlukan pendefinisian critical section dan memaksakan mutual exclusion diantara proses-proses kongkuren yang sedang berjalan. Pemaksaan mutual exclusion merupakan landasan pemrosesan kongkuren, namun pemaksaan mutual exclusion dapat menimbulkan 2 masalah yaitu :

- **Deadlock**
  Adalah banyak proses yang saling menunggu hasil dari proses yang lain untuk dapat melanjutkan atau menyelesaikan tugasnya.
  Contoh :
  - 2 proses : P1 dan P2
    Dua sumber daya kritis, R1 dan R2
    Proses P1 dan P2 harus mengakses kedua sumber daya
    Skenario berikut dapat terjadi : R1 diberikan ke P1 sedang R2 diberikan ke P2
    Karena untuk melanjutkan eksekusi memerlukan kedua sumberdaya sekaligus maka kedua proses akan saling menunggu sumber daya lain selamanya. Kedua proses dalam kondisi deadlock, tidak dapat membuat kemajuan apapun.
- **Startvation**
  Adalah suatu proses akan menunggu suatu kejadian atau hasil suatu proses lain supaya dapat menyelesaikan tugasnya, tetapi kejadian yang ditunggu tidak pernah terjadi karena selalu diambil lebih dulu oleh proses yang lain.
  Contoh :
  - Tiga proses : P1, P2, P3
    P1,P2,P3 memerlukan pengaksesan sumber daya R secara periodik
    Skenario berikut dapat terjadi :
    - P1 diberi sumberdaya R, P2 dan P3 blocked menunggu sumber daya R
    - Ketika P1 keluar dari critical section, P2 dan P3 diijinkan mengakses R
    - Asumsi P3 diberi hak akses, kemudian setelah selesai hak akses kembali diberikan ke P1 yang saat itu kembali membutuhkan sumberdaya R.

    Jika pemberian hak akses terus-menerus antara P1 dan P3, maka P2 tidak pernah memperoleh pengaksesan sumberdaya R, meski tidak ada deadlock. Pada situasi ini P2 mengalami startvation

Pada sistem dengan banyak proses (sistem kongkuren) terdapat 3 kategori interaksi diantara proses-proses, yaitu :

Kategori interaksi	Hubungan	Pengaruh pada proses yang lain	Masalah pengendalian
Proses-proses saling tidak peduli (independen)	Persaingan Contoh : 2 proses berusaha mengakses printer yg sama	• hasil dari 1 proses tidak berpengaruh terhadap proses yang lain • pewaktuan proses dapat berpengaruh	• mutual exclusion • deadlock • startvation
Proses-proses saling peduli secara tidak langsung	Kerjasama dengan pemakaian bersama	• hasil dari 1 proses dapat tergantung dari informasi yang dihasilkan proses lain • pewaktuan proses dapat berpengaruh	• mutual exclusion • deadlock • startvation • koherensi data (keterhubungan data)
Proses saling peduli secara langsung	Kerjasama dengan komunikasi	• hasil dari 1 proses dapat tergantung dari informasi yang dihasilkan proses lain • pewaktuan proses dapat berpengaruh	• deadlock • startvation

Race Condition

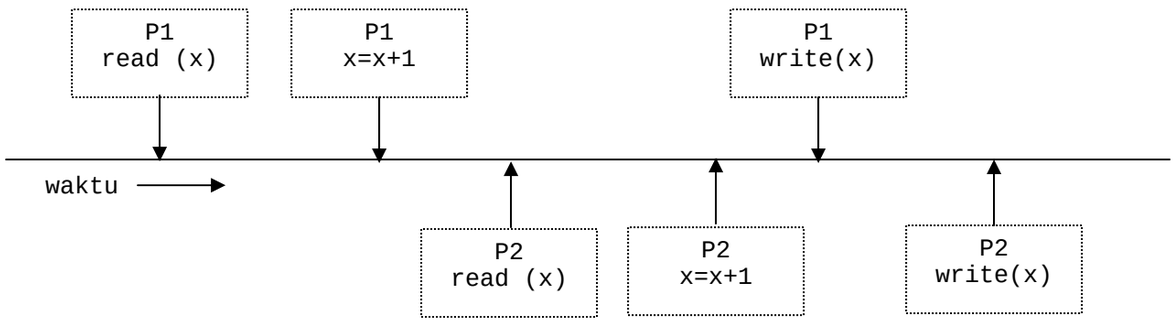
Merupakan sebuah kondisi dimana 2 atau lebih proses membaca atau menulis data/variabel yang digunakan bersama,dan hasilnya tergantung dari proses mana yang terakhir menggunakan data tersebut.

Contoh 1 : sebuah algoritma yang akan diakses oleh 2 proses P1 dan P2

```
read (x)
x=x+1
write (x)
```

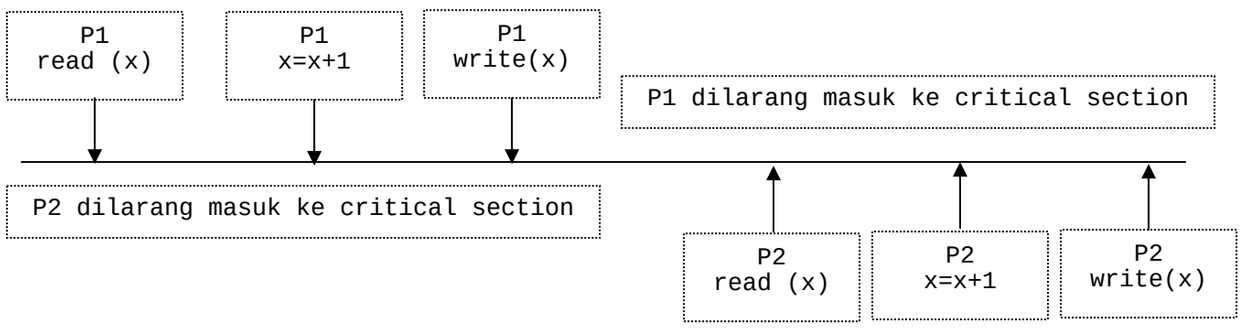
critical section

Masalah terjadi ketika P1 membaca input variabel yang digunakan bersama, yaitu x (merupakan critical section) dan mulai operasi penambahan, kemudian P2 juga membaca input variabel tersebut sebelum P1 dapat menampilkannya ke layar monitor.



- Proses P1 membaca nilai variabel x yang diinputkan, misal : 3
- Proses P1 menjalankan operasi penambahan sehingga x bertambah 1, menjadi 4
- Proses P2 membaca nilai variabel x yang diinputkan, misal : 5
  Terjadi masalah untuk P1 karena variabel x yang seharusnya 4 menjadi 5
- Proses P2 menjalankan operasi penambahan sehingga x bertambah 1, menjadi 6
- Proses P1 mencetak variabel x yang hasilnya adalah 6 → merupakan nilai yang salah bagi P1, seharusnya 4.
- Proses P2 mencetak variabel x yang hasilnya adalah 6 → merupakan nilai yang benar bagi P2.

Untuk mengatasi race condition adalah dengan meyakinkan bahwa hanya ada 1 proses saja yang akan mengeksekusi critical section. Dengan kata lain, proses akan mengeksekusi critical section secara sekuensial (berurutan), tidak secara paralel (bersamaan) → disebut mutual exlusion.



Mutual exclusion dapat dikerjakan dengan membuat eksekusi seluruh critical section menjadi eksekusi tunggal di data / variabel yang digunakan bersama sehingga tidak dapat dikerjakan oleh proses yang lain pada saat yang sama. Eksekusi tunggal ini disebut dengan **atomic action**.

Contoh 2 :

Pada aplikasi tabungan, misal rekening A berisi Rp. 1 juta terdaftar di kantor cabang Semarang. Pada suatu saat program aplikasi di kantor cabang Jakarta melayani penyetoran Rp.3 juta ke rekening tersebut. Program aplikasi membaca saldo akhir rekening A. Pada waktu yang hampir

bersamaan di kantor cabang Semarang juga terjadi transaksi yaitu penyetoran Rp. 5 juta ke rekening A. Program aplikasi di Semarang membaca saldo masih Rp. 1 juta.
Beberapa skenario dapat terjadi bila mutual exclusion tidak terjamin :

- Program aplikasi di Semarang dilakukan secara cepat menulis ke rekening A sehingga dihasilkan Rp. 6 juta. Setelah itu program aplikasi di kantor cabang Jakarta menimpa hasil tersebut dengan Rp. 4 juta. Hasil akhir yang diperoleh adalah Rp. 4 juta yang seharusnya Rp. 9 juta.
- Program aplikasi di Jakarta dilakukan secara cepat menulis ke rekening A sehingga dihasilkan Rp. 4 juta. Setelah itu program aplikasi di cabang Semarang menimpa hasil tersebut dengan Rp. 6 juta. Hasil akhir yang diperoleh adalah Rp. 6 juta yang seharusnya Rp. 9juta.

Masalah ini menjadi bahasan sistem operasi dan pembuat sistem manajemen basis data.

Mutual exclusion dapat terjadi jika memenuhi syarat / kriteria sebagai berikut :

a. Mutual exclusion harus dijamin, hanya ada 1 proses dalam 1 waktu yang diijinkan masuk ke critical section.
b. Proses yang berada di noncritical section, dilarang mem-blocked proses-proses lain yang ingin masuk critical section..
c. Harus dijamin proses yang ingin masuk critical section tidak menunggu selama waktu yang tak berhingga. Tidak boleh terdapat deadlock atau startvation.
d. Ketika tidak ada proses pada critical section maka proses yang ingin masuk critical section harus diijinkan masuk tanpa waktu tunggu.
e. Tidak ada asumsi mengenai kecepatan relatif proses atau jumlah proses yang ada
f. Proses berada di critical section dengan waktu yang terbatas.

Metode yang diusulkan untuk menjamin mutual exclusion antara lain :

1. **Metode variabel lock**
   Ketika proses hendak masuk critical section, lebih dulu memeriksa variabel lock. Jika variabel lock berisi 0, proses menset variabel lock menjadi satu dan kemudian masuk critical section. Tetapi jika variabel lock berisi 1 berarti terdapat proses lain pada critical section sehingga proses menunggu sampai nilai variabel lock menjadi nol.

2. **Metode bergantian secara ketat**
   Metode ini mengamsumsikan dapat menggilir masuk critical section secara bergantian terus menerus. Metode ini melakukan inspeksi terhadap variabel yang berfungsi untuk memasuki critical section.

3. **Metode dengan Busy Waiting**
   Busy waiting adalah adanya proses yang sibuk menunggu dan tidak mengerjakan apapun sampai mendapatkan ijin untuk masuk ke critical section. Sibuk menunggu disini berarti bahwa proses tersebut sedang menunggu dan terus menerus sibuk memeriksa status ijinnya sehingga memakan waktu prosesor.

   Metode dengan dukungan perangkat lunak

   a. **Algoritma Dekker**
      Mempunyai ciri – ciri :
   - Tidak memerlukan instruksi-instruksi perangkat keras khusus
   - Proses yang beroperasi di luar critical section tidak dapat mencegah proses lain yang ingin masuk critical section.
   - Proses yang ingin masuk critical section akan segera masuk bila dimungkinkan.

   b. **Algoritma Peterson**
      Lebih sederhana dan lebih baik daripada algoritma Dekker.

   Metode dengan dukungan perangkat keras

   a. **Pematian Interrupt**
      Pada mesin berprosesor tunggal, proses yang bersamaan tidak dapat dioverlap (secara nyata bersamaan), hanya dapat diinterleave(berjarak). Oleh karena itu untuk menjamin mutual exclusion, proses tidak boleh diinterupt ketika sedang berada di critical section. Tetapi hal ini mengakibatkan efisiensi eksekusi proses menurun, karena kemampuan prosesor terbatas untuk menginterleave program. Masalah lainnya adalah cara ini tidak dapat berjalan dalam arsitektur multiprosesor karena jika terdapat dua prosesor atau lebih, mematikan interupt hanya berpengaruh pada pemroses (prosesor) yang mengeksekusi intruksi itu. Pematian interupt tidak mempengaruhi prosesor lain. Prosesor lain masih bebas memasuki critical section yang sedang dimasuki proses lain.

   b. **Menggunakan Instruksi Test and Set Lock (TSL)**
      Banyak arsitektur terutama yang didesain untuk menggunakan multiprosesor, menyediakan instruksi khusus untuk memeriksa (test) dan memodifikasi isi dari lokasi. Pada saat instruksi ini dijalankan maka CPU akan mengunci saluran bus untuk memori, untuk mencegah prosesor lain mengakses lokasi memori tersebut.

   c. **Menggunakan Instruksi Mesin yang Khusus (Instruksi Exchange / XCHG)**
      Pada tingkatan hardware, suatu akses ke suatu lokasi memori meniadakan akses yang lain ke lokasi memori yang sama. Dengan dasar itu para perancang prosesor membuat beberapa instruksi mesin yang membawa 2 action secara atomic yaitu *membaca dan menulis* atau *menulis dan menguji lokasi memori* tunggal dengan satu instruksi fetch cycle. Karena action ini diadakan dalam siklus instruksi tunggal maka tidak akan dicampuri oleh instruksi yang lain.
      Keuntungan:
   - dapat diterapkan ke sembarang jumlah proses dalam prosesor tunggal/multiprosesor yang berbagipakai memori utama
   - sederhana dan mudah diperiksa
   - dapat digunakan untuk mendukung banyak critical section; setiap critical section dapat didefinisikan oleh variabelnya sendiri.

Kerugian :

- ketika proses menunggu untuk mengakses critical section, maka proses akan selalu memakan waktu prosesor
- ketika sebuah proses meninggalkan critical section dan lebih dari satu proses menunggu, pemilihan proses yang akan masuk ke critical section tidak jelas, maka akan ada suatu proses yang selalu dalam status menunggu dan tidak pernah dapat masuk ke critical section → disebut startvation
- pada prosesor tunggal, dimisalkan proses P1 mengeksekusi instruksi khusus dan masuk ke critical section. P1 kemudian diinterupt dan memberikan prosesor ke P2 yang mempunyai prioritas lebih tinggi. Disini P1 masih menggunakan sumber dayanya. Jika P2 berusaha menggunakan sumber daya yang sama dengan P1, maka P2 akan ditolak karena mekanisme mutual exclusion sehingga P2 akan masuk ke looping busy waiting. P1 tidak akan pernah dieksekusi lagi karena memiliki prioritas yang lebih rendah dibandingkan proses yang siap, yaitu P2, sehingga terjadi peristiwa saling tunggu abadi → deadlock.

4. Metode dengan Semaphore

Untuk berbagi pakai sumber daya, proses harus memecahkan masalah mutual exclusion. Sedangkan untuk bekerja sama dalam suatu tugas, proses harus masalah pen-signal-an. Oleh karena itu muncul variabel sederhana yang disebut semaphore.
Prinsip dasar semaphore adalah dua atau lebih proses bekerja sama dengan signal yang sederhana,misal proses dipaksa untuk berhenti di tempat yang ditunjuk sampai menerima suatu signal khusus. Untuk mengirim signal lewat semaphore, proses mengeksekusi signal yang sederhana. Untuk menerima signal lewat semaphore, proses mengeksekusi operasi **wait()** yang sederhana. Jika signal yang bersangkutan belum dikirimkan, maka proses ditunda sampai pengiriman datang.

Semaphore dapat dianalogikan dengan sebuah kunci yang tergantung pada gantungan. Ketika proses memanggil **wait()**; hal ini berarti kita sedang mencari kunci di gantungan. Jika kunci ada di gantungan, maka kunci diambil dan pemanggil **wait()** akan melanjutkan tugasnya. Jika kunci tersebut tidak ada, maka pemanggil **wait()** akan menunggu di sekitar gantungan sampai kunci dipasangkan kembali ke gantungan. Pemanggil **signal()** yang akan menempatkan kunci kembali di gantungan.

Jadi ada dua operasi yang didefinisikan dalam semaphore :

- semaphore dapat diinisialisasi dengan nilai non-negatif
- **operasi down / operasi wait()** mengurangkan nilai semaphore. Jika nilai semaphore menjadi negatif maka proses yang mengeksekusi wait() diblok
- **operasi signal/ operasi signal()** menambahkan nilai semaphore. Jika nilai semaphore menjadi negatif maka proses yang diblok oleh operasi wait() dibebaskan (tidak diblok lagi)

Mutual exclusion dengan semaphore

Sebelum masuk ke critical section, proses melakukan Down . Bila berhasil maka proses masuk critical section. Bila tidak berhasil maka proses di-block pada semaphore itu. Proses yang diblock akan dapat melanjutkan kembali bila proses di critical section keluar dan melakukan operasi Up sehingga menjadikan proses yang diblock ready dan berlanjut hingga operasi Down berhasil.

Pematian interupt dengan semaphore

Sistem operasi mematikan semua interupt selagi memeriksa semaphore, memperbarui, dan menjadikan proses diblock. Karena semua aksi hanya memerlukan beberapa instruksi, pematian interupt tidak merugikan.

Instruksi TSL dengan semaphore

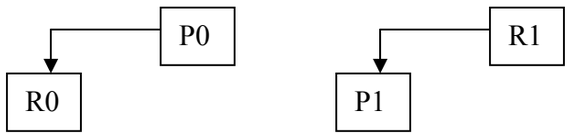
Pada banyak pemroses, tiap semaphore dilindungi variabel lock dan instruksi tsl agar menjamin hanya satu pemroses yang saat itu memanipulasi semaphore.

VI. DEADLOCK

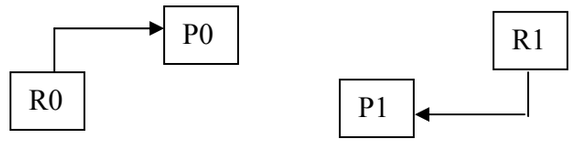
Deadlock : jika proses menunggu suatu kejadian tertentu yang tak akan pernah terjadi.
Sekumpulan proses berkondisi deadlock bila setiap proses yang ada di kumpulan menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu.
Proses menunggu kejadian yang tak akan pernah terjadi.

Urutan kejadian pengoperasian peralatan masukan / keluaran :
- meminta (request) : meminta pelayanan perangkat masukan/keluaran
- memakai (use) : memakai perangkat masukan / keluaran
- melepaskan (release) : melepaskan pemakaian perangkat masukan/keluaran

Model deadlock 2 proses dan 2 sumber daya
Misal : 2 proses P0 dan P1
2 sumber daya R0 dan R1
P0 meminta sumberdaya R0. Sumber daya R1 dialokasikan ke P1.

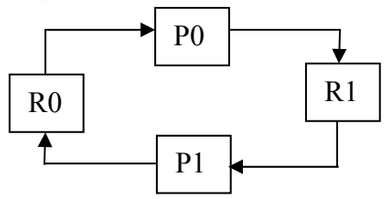


Skenario yang menimbulkan deadlock :
P0 dialokasikan R0
P1 dialokasikan R1



Kemudian
P0 sambil masih menggenggam R0, meminta R1
P1 sambil masih menggenggam R1, meminta R0

Terjadi deadlock karena sama-sama akan saling menunggu. Terjadinya deadlock ditandai munculnya graph melingkar.



Deadlock dapat terjadi dengan melibatkan lebih dari 2 proses dan 2 sumber daya.

Syarat terjadinya deadlock :
1. mutual exclusion (mutual exclusion conditional)
   tiap sumber daya saat itu diberikan pada tepat 1 proses.
2. kondisi genggam dan tunggu (hold and wait)
   proses-proses yang sedang menggenggam sumber daya, menunggu sumber daya-sumber daya baru.
3. kondisi non-preemption (non-preemption condition)
   sumber daya-sumber daya yang sebelumnya diberikan tidak dapat diambil paksa dari proses itu.Sumber daya harus secara explisit dilepaskan dari proses yang menggenggamnya.
4. kondisi menunggu secara sirkuler (circular wait condition)
   harus terdapat rantai sirkuler dari dua proses atau lebih, masing-masing menunggu sumber daya yang digenggam oleh anggota berikutnya pada rantai itu.

Terjadi deadlock bila terdapat ketiga kondisi itu, tetapi adanya ketiga kondisi itu belum berarti terjadi deadlock. Deadlock benar-benar terjadi bila syarat keempat terpenuhi. Kondisi keempat merupakan keharusan bagi terjadinya deadlock.

Deadlock bisa terjadi pada saat proses akan mengakses objek seperti file,device, dll secara tidak semestinya. Objek tersebut dinamakan **resource.**

Terdapat 2 jenis resource :

- **preemtable resource**
  resource yang dapat diambil dan dilepas dari proses yang sedang memakainya tanpa memberikan efek apapun pada proses tersebut.
- **Non-preemtable resource**
  resource tidak dapat diambil dari proses yang sedang membawanya, karena akan mengakibatkan kegagalan komputasi, contoh : printer, bila suatu proses sedang menggunakan printer untuk mencetak, maka proses lain tidak dapat menggunakan printer tersebut.

Metode mengatasi deadlock

1. **Metode pencegahan deadlock**
   - Tiap proses harus meminta semua sumber daya yang diperlukan secara sekaligus dan tidak berlanjut sampai semuanya diberikan.
   - Jika proses sedang memegang sumberdaya tertentu, untuk permintaan berikutnya proses harus melepas dulu sumberdaya yang dipegangnya.
   - Beri pengurutan linier terhadap tipe-tipe sumber daya pada semua proses, yaitu jika proses telah dialokasikan suatu tipe sumber daya, proses hanya boleh meminta tipe sumber daya pada urutan berikutnya.

2. **Penghindaran deadlock**
   Menghindari deadlock dengan cara hanya memberi akses ke permintaan sumber daya yang tidak mungkin menimbulkan deadlock.
   - jika pemberian akses sumber daya tidak mungkin menuju deadlock, sumber daya diberikan ke peminta.
   - Jika tidak aman (memungkinkan timbulnya deadlock), proses yang meminta di suspend sampai suatu waktu permintaannya aman diberikan. Kondisi ini biasanya terjadi setelah 1 sumber daya / lebih yang semula dipegang oleh proses-proses aktif lain dilepaskan.

   Agar dapat mengevaluasi safe-nya state sistem, penghindaran deadlock mengharuskan semua proses menyatakan jumlah kebutuhan sumber daya maksimum sebelum eksekusi. Begitu eksekusi dimulai, tiap proses meminta sumber daya saat diperlukan sampai batas maksimum yang dinyatakan di awal. Proses-proses yang menyatakan kebutuhan sumber daya melebihi kapasitas total sistem tidak dapat dieksekusi.

State selamat dan state tidak selamat

State selamat (safe state)

State selamat (safe state) : jika tidak deadlock dan terdapat cara untuk memenuhi semua permintaan yang ditunda tanpa menghasilkan deadlock dengan menjalankan proses secara hati-hati mengikuti suatu urutan tertentu.

Contoh :

Sistem dengan 10 sumber daya setipe, proses A memerlukan sumber daya maksimum sebanyak 10, sedang saat ini menggenggam 2 sumber daya. proses B memerlukan sumber daya maksimum sebanyak 3, sedang saat ini menggenggam 1 sumber daya. Proses C memerlukan sumber daya maksimum sebanyak 7, sedang saat ini menggenggam 3 sumber daya. Masih tersedia 4 sumber daya.

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	1	3
C	3	7
Tersedia	4	

State dinyatakan selamat (safe) karena terdapat barisan pengalokasian yang dapat memungkinkan semua proses selesai. Dengan penjadwalan secara hati-hati, sistem dapat terhindarkan dari deadlock. Barisan tersebut adalah :

Langkah 1 :
Alokasikan semua sumberdaya ke proses C, tunggu sampai proses C selesai.

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	1	3
C	7	7
Tersedia	0	

Setelah proses C selesai menjadi :

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	1	3
C	0	-
Tersedia	7	

Langkah 2 :
Alokasikan 2 sumberdaya ke proses B, tunggu sampai proses B selesai.

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	3	3
C	0	-
Tersedia	5	

Setelah proses B selesai menjadi :

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	0	-
C	0	-
Tersedia	8	

Langkah 3 :
Alokasikan 8 sumberdaya ke proses A, tunggu sampai proses A selesai.

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	10	10
B	0	-
C	0	-
Tersedia	0	

Setelah proses A selesai menjadi :

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	0	-
B	0	-
C	0	-
Tersedia	10	

Ketiga proses tersebut dengan penjadwalan yang hati-hati dapat menyelesaikan proses mereka dengan sempurna.

State tak selamat (unsafe state)
State tak selamat (unsafe state) : jika tidak terdapat cara untuk memenuhi semua permintaan yang saat ini ditunda dengan menjalankan proses-proses dengan suatu urutan.
Contoh :
State dibawah ini sama dengan state selamat sebelumnya, tapi dapat berubah menjadi state tak selamat bila alokasi sumber daya tak terkendali.

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	2	10
B	1	3
C	3	7
Tersedia	4	

State berubah menjadi tak selamat bila 2 permintaan sumber daya oleh proses A dilayani kemudian permintaan 1 sumber daya oleh proses B dilayani. Maka state menjadi :

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	4	10
B	2	3
C	3	7
Tersedia	1	

Alokasikan 1 sumberdaya ke proses B, tunggu sampai proses B selesai.

Proses	Jumlah sumberdaya Digenggam	Maksimum sumberdaya dibutuhkan
A	4	10
B	3	3
C	3	7
Tersedia	0	

Setelah proses B selesai menjadi :

Proses	Jumlah sumberdaya digenggam	Maksimum sumberdaya dibutuhkan
A	4	10
B	0	-
C	3	7
Tersedia	3	

Sumber daya yang tersedia tinggal 3 sedangkan 2 proses yang sedang aktif masing-masing membutuhkan 6 dan 4 sumber daya.

State tak selamat bukan berarti deadlock, tetapi state tersebut berkemungkinan menuju deadlock.

3. Deteksi dan pemulihan deadlock

Deteksi deadlock adalah teknik untuk menentukan apakah deadlock terjadi serta mengidentifikasi proses-proses dan sumberdaya-sumberdaya yang terlibat deadlock. Periode yang biasa dilakukan adalah memonitor permintaan dan pelepasan sumber daya.
Bila sistem terdapat deadlock maka deadlock harus diputuskan. Biasanya beberapa proses akan kehilangan sebagian atau semua kerja yang telah dilakukan. Hal ini lebih baik daripada terjadinya deadlock yang berarti semua proses tidak menghasilkan apapun.

Teknik pemulihan yang biasa digunakan adalah menghilangkan (suspend / kill) proses-proses dari sistem dan pengklaiman kembali sumberdaya yang dipegang proses-proses tersebut. Proses yang dihilangkan biasanya cacat tetapi proses lain dapat menyelesaikan prosesnya. Pendekatan-pendekatan berikut dapat dilakukan untuk pemulihan deadlock :

- singkirkan semua proses yang terlibat deadlock
- backup semua proses yang terlibat deadlock ke suatu check point yang didefinisikan sebelumnya dan jalankan kembali proses itu.
- secara berurutan abaikan proses-proses sampai deadlock tidak terjadi lagi. Urutan proses yang dipilih untuk disingkirkan berdasar kriteria ongkos minimum.
- secara berurutan preempt sumberdaya-sumberdaya sampai tidak ada deadlock lagi. Proses yang kehilangan sumber daya karena preemption harus dikembalikan (roll-back) ke titik sebelum memperoleh sumber daya.

Kriteria pemilihan proses yang akan disingkirkan (suspend/kill) :

- waktu pemrosesan yang dijalankan paling kecil
- jumlah baris keluaran yang diproduksi paling kecil

- mempunyai estimasi sisa waktu eksekusi terbesar
- jumlah total sumber daya terkecil yang telah dialokasikan
- prioritas terkecil.

Dalam penyingkiran proses, sistem harus mengembalikan berkas-berkas yang telah dimodifikasi oleh proses-proses yang disingkirkan ke keadaan asli karena berpengaruh terhadap konsistensi data sistem itu.